REMARKS/ARGUMENTS

Reconsideration of the application is requested.

Claims 1 and 3-12 remain in the application. Claims 1 and 11 have been amended. Claims 2 and 3 have been canceled.

In item 6 on page 2 of the above-identified Office Action, claim 11 has been rejected as failing to comply with the enablement requirement under 35 U.S.C. § 112, first paragraph.

More specifically, the Examiner states that the limitation " a process is called by assigning a process number, a priority, and a memory address of a starting point of the process in the program memory" in claim 11 is not consistent with the description on page 15 of the instant specification.

Claim 11 has been revised to state that the process is called "by **executing a run instruction** assigning a process number" (emphasis added), which is consistent with the description in the instant specification.

Support for the change may be found on page 15, lines 8-11 of the specification of the instant application.

Claim 11 is believed to be consistent with and supported by the instant specification. Therefore, the Examiner is requested to withdraw the rejection based on 35 U.S.C. § 112, first paragraph.

It is accordingly believed that claim 11 meets the requirements of 35 U.S.C. § 112, first paragraph. The above noted changes to the claims are provided solely for clarification or cosmetic reasons.

In item 8 on page 3 of the above-identified Office Action, claims 1-8 and 10 have been rejected as being anticipated by Gupta et al. (U.S. Patent No. 5,941,983) (hereinafter "Gupta") under 35 U.S.C. § 102(a).

Dependent claim 9 has not been included in the aforesaid rejection, however, claim 9 is mentioned in item 17 as if the Examiner intended to reject the claim under 35 U.S.C. § 102(a). However, the Examiner's explanation in items 17a-g of the basis of the rejection states that Gupta does not "explicitly" disclose several claimed features and proceeds to describe in great detail where the Examiner believes Gupta "inherently" shows the features. Applicants respectfully submit that if the Examiner has to go to such great lengths to

explain where the claimed limitations are shown in Gupta then the basis for the rejection of claim 9 should be under 35 U.S.C. § 103(a), not 35 U.S.C. § 102(a). In any event, claim 9 will be treated as if it were included in the aforesaid rejection.

The rejection has been noted and the claims have been amended in an effort to even more clearly define the invention of the instant application. Support for the changes is found in original claims 2-3 and on page 15, line 23, page 16, lines 9-16, page 17, lines 3-4, and page 17, lines 7-10 of the specification of the instant application.

Before discussing the prior art in detail, it is believed that a brief review of the invention as claimed, would be helpful. Claim 1 calls for, *inter alia*, a data-processing device **for processing in parallel a plurality of independent processes**, having:

a program memory having stored therein at least one compiled program **with a multiplicity N of independent processes**, the compiled program including information on parallelism and **including a multiplicity of bundles with a plurality of**

instructions of a process, the instructions of a bundle being executable in parallel;

a branching control unit connected to and addressing said program memory;

a register for storing flags and data which are switched in dependence on a process being executed;

and controlling an output of instructions to be processed in parallel in dependence on information contained in the instructions and included in a compiling time of the program;

a number N of instruction buffers being connected in parallel downstream of said program memory for storing instructions read out from said program memory, an instruction bundle being read into one of said instruction buffers and a second instruction bundle associated with a different process being read into another one of said instruction buffers; and

an instruction output selector being connected to and controlled by said process flow control unit **causing said instruction output selector to read out instructions from said instruction buffers and output N instructions in parallel,**

said instruction output selector having a multiplexer logic and selecting one of either one instruction from a first instruction buffer and one instruction from a second instruction buffer, or two instructions from one of said first and second instruction buffers. (emphasis added)

Gupta discloses a method for executing instructions out-of-order to improve performance of a processor. The method includes compiling the instructions of ***a single program*** into separate queues along with various encoded dependencies between instructions in the different queues. The processor then issues instructions from each of these queues independently, except that the processor enforces the encoded dependencies among instructions from different queues.

However, during execution of the program, the processor in Gupta "issues instructions from each queue **in sequential order**" (col. 6, lines 15 and 16) and not "in parallel" as recited in claim 1 of the instant application. Moreover, Gupta further discloses that the processor can issue instructions from each queue independently and as fast as a functional unit can handle new instructions **as long as the dependencies are satisfied** (Col. 6, lines 26 to 29).

In contrast, the present invention as claimed in independent claim 1, provides a data-processing device wherein the data-processing device or processor can carry out/process ***a plurality of independent processes*** (or threads) in parallel.

Accordingly, the claimed invention benefits from the fact that the dependencies of data and instructions in independent processes running in parallel are typically significantly smaller than in an individual program. For example, the single sequential program flow as described in Gupta (see e.g., col. 6, lines 15-16) has more dependencies of data and instructions for parallel processing than when at least two independent processes are processed in parallel as described in the instant application. More specifically, in the instant application the instructions, which are to be processed in parallel, are fetched from a program memory according to a clock cycle and each individual parallel process is assigned a priority.

While in Gupta the queues may be considered equal to the independent processes, Gupta does not disclose a program having independent processes includes a multiplicity of bundles with a plurality of instructions of a process, as recited in claim 1. Further, Gupta does not disclose that the

instructions of a bundle can be executed in parallel, as recited in claim 1.

Claim 1 also recites that the instruction output selector has a multiplexer logic and that the instruction selector selects either one instruction from a first instruction buffer and one instruction from a second instruction buffer, or two instructions from one of the two instruction buffers. This feature is not shown or taught in Gupta. To the contrary, in Gupta each separate queue is linked to one associated instruction fetch unit , as illustrated by 80 in Fig. 3.

Therefore, it is apparent that Gupta does not show the claimed features of the present invention as recited in claim 1.

A major advantage of the present claimed invention is that not only the independent processes can be executed in parallel but also instructions within one bundle can be executed in parallel. This advantage can be further explained with reference to Fig. 1 of the instant application. A first bundle of a first process is loaded into the instruction buffer (13) and a second bundle of a second process is loaded into the second instruction buffer (14). It is known that all instructions of the first bundle are independent with regard

to the instructions of the second bundle, because the first bundle belongs to the first process and the second bundle belongs to the second process and both processes are independent, as recited in claim 1. In one case, the instruction issue selector (15) selects from each of the two instruction buffers (13, 14) one instruction, so these instructions can be fed to the execution units (19, 20) and thus be processed in parallel. In this scenario both the first and the second processes are executed in parallel and both execution units (19, 20) are under full load. In case an instruction of the first process requests a memory access this instruction cannot be executed immediately by the execution unit (19). Additionally, further instructions of the first process may be not executable in parallel to the previously mentioned instruction. Thus, the execution unit (19) would be without load. The advantage of the present invention is that in case a bundle of two or more instructions is present in the second instruction buffer (14), two of these instructions could be loaded by the instruction issue selector (15) and subsequently executed in parallel by both execution units (19,20). Thus, some load for the execution unit (19) may be maintained even if no instructions of the first process can be executed at the moment.

The reference numbers used above are for explanation purposes only and are not intended to be limiting in anyway of the claimed invention.

Neither Gupta, or any of the secondary references, or any combination thereof, show or teach being able to achieve a permanent load of the execution units by using bundles and parallel processes.

Clearly, Gupta does not show "a data-processing device for processing in parallel a plurality of independent processes bundles with a plurality of instructions of a process, the instructions of a bundle being executable in parallel" and a "program flow control unit controlling a fetching of bundles to be processed in parallel from said program memory, controlling said branching control unit, and controlling an output of instructions to be processed in parallel" and "said instruction output selector having a multiplexer logic and selecting one of either one instruction from a first instruction buffer and one instruction from a second instruction buffer, or two instructions from one of said first and second instruction buffers" as recited in independent claim 1 of the instant application.

In item 21 on page 9 of the above-identified Office Action, claim 11 has been rejected as being unpatentable over Gupta in view of Ito et al. (U.S. Patent No. 5,742,782) (hereinafter "Ito") under 35 U.S.C. § 103(a).

The previous discussion of Gupta is equally applicable in the rejection.

Ito does not overcome the deficiencies of Gupta. Ito discloses a processing apparatus for executing a plurality of **VLIW threads** in parallel. The processing apparatus simultaneously executes multiple threads of long instructions, where each thread is made up of multiple operational instructions. The gain by virtue of using ILP is, however, restricted by the inherent dependencies of the data operations and control operations. In order to avoid such dependencies, complex preprocessing - for example taking into account data and control operation dependencies during the actual programming - is necessary and this in turn makes the entire development process more expensive.

Ito does not assign "a process number, a priority, and a memory address ... in the program memory" as recited in claim 11 of the instant application. Rather Ito develops these signals

during execution and changes the cited signals during the course of execution. For example, FIG. 13 indicates that the TNC/Signal155 or alleged "priority" changes in block (6) and block (4). Thus, TNC appears to be more like a counter or queue position than a priority. In contrast, FIG. 4 of the instant application shows the state diagram of the flow control unit 10, in which the priority of process A is higher than the priority of process B.

Clearly, Ito does not show "**assigning** a process number, a **priority**, and a memory address of a starting point of the process in the program memory" as recited in claim 11 of the instant application. Nor does Ito show or suggest "**a process is called with a run instruction assigning** a process number, a priority, and a memory address of a starting point of the process in the program memory" as recited in claim 11.

None of the prior art references of Gupta and Ito individually or any combination thereof show the claimed limitations of claim 11.

In item 24 on page 9 of the above-identified Office Action, claim 12 has been rejected as being unpatentable over Gupta in

view of Allen et al. (U.S. Patent No. 6,404,752) (hereinafter "Allen") under 35 U.S.C. § 103(a).

The previous discussion of Gupta is equally applicable in the rejection.

Allen does not overcome the deficiencies of Gupta. Allen discloses a network switch in which data flow handling and flexibility is enhanced by a network processor capable of cooperating with other elements, such as an optional switching fabric device, to execute instructions that direct the flow of data within a network. The network processor includes a plurality of cooperating interface processors and various peripheral elements **formed on a semiconductor substrate**. Allen also discloses, "network processors can increase bandwidth and solve latency problems in a broad range of application by allowing networking tasks previously handled in software to be executed in hardware" (col. 2, lines 64-67). However, the combination of Allen and Gupta would require using **multiple network processors**, where each processor may process a single process in parallel. Unfortunately, this would require a tremendous amount of processing overhead to coordinate parallel processing and indicates that there is not

a reasonable expectation of success for the proposed combination.

Clearly, Allen does not show "a data-processing device for processing in parallel a plurality of independent processes" as recited in dependent claim 12 of the instant application. Nor does Allen show or suggest "a network processor for processing layer 1 to 7 of protocol stacks in applications including LAN, ATM switches, IP routers, and frame relays based on a system selected from the group consisting of DSL, Ethernet, and cable modems" as recited in claim 12 of the instant application.

None of the prior art references of Gupta or Allen individually or in combination show the claimed limitations of claim 12.

It is accordingly believed to be clear that none of the references, whether taken alone or in any combination, either show or suggest the features of claim 1. Claim 1 is, therefore, believed to be patentable over the art. The dependent claims are believed to be patentable as well because they all are ultimately dependent on claim 1.

In view of the foregoing, reconsideration and allowance of claims 1 and 4-12 are solicited.

In the event the Examiner should still find any of the claims to be unpatentable, counsel would appreciate receiving a telephone call so that, if possible, patentable language can be worked out. In the alternative, the entry of the amendment is requested, as it is believed to place the application in better condition for appeal, without requiring extension of the field of search.

If an extension of time for reply to this paper is required, petition for extension is herewith made.

Please charge any other fees that might be due with respect to Sections 1.16 and 1.17 to the Deposit Account of Lerner and Greenberg, P.A., No. 12-1099.

Respectfully submitted,

F. Donald Paris Reg. No. 24,054

FDP/bb

September 30, 2004

Lerner and Greenberg, P.A.
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100
Fax: (954) 925-1101